Attachment G – Requirements Matrix

Req. No.	Requirement	Release 4.1 Impl.	Comments
AR	***Architectural Requirements***		
AR1	The SIMSS shall have a client/server architecture with the client providing the user interface and the server providing the functionality.	Full	
AR2	The SIMSS shall be capable of using multiple clients and servers.	Partial	
AR2.1	The SIMSS shall be capable of interfacing with multiple servers from a single client.	Full	
AR2.2	The SIMSS shall be capable of interfacing with multiple clients from a single server. Only one client shall be allowed to control the server with the other clients providing a display-only capability.		
AR2.3	The SIMSS shall be capable of providing password protection for invoking or transferring the master user interface.		
AR3	The SIMSS shall be able to run client and server on same or separate hosts.	Full	
AR4	Any SIMSS client shall be capable of running with any SIMSS server.	Full	
AR5	The SIMSS client shall be capable of running on any pure Java compliant virtual machine.	Full	
AR6	The SIMSS server shall be capable of running on a machine with a Windows NT operating system.	Full	
AR7	The SIMSS server shall be capable of running on a machine with a Linux operating system.		
AR8	The SIMSS shall provide remote access to the SIMSS server for operations.		
AR9	The SIMSS shall use TBD security standards for remote access.		
AR10	The SIMSS shall consist of a collection of independent modules capable of being connected together via links for a specific function.	Full	
AR10.1	The modules shall provide a standard user and programmatic interface.	Full	
AR10.2	Each module shall have a client and a server component.	Full	
AR10.3	Each module shall provide a specific, logically distinct element of overall SIMSS functionality.	Full	
AR11	The SIMSS shall be capable of creating and	Full	

	running one or more configurations (projects) under operator control. A project is a collection of SIMSS modules and links intended to perform a specific function.		
AR12	The output channel of any SIMSS module shall be able to interface with the input channel of any other SIMSS module, and vice versa.	Partial	See limitations (Attachment H of R4.1 delivery package)
AR13	The SIMSS shall be capable of being a component of a larger simulation system that is IEEE-1516 (DMSO HLA) compliant.		

UI	***User Interface Requirements***	***R4.1 Impl.***	***Comments***
UI1	The SIMSS shall provide a graphical user interface.	Full	
UI1.1	The SIMSS shall provide a user interface that can be run within a Web browser.		
UI1.2	The SIMSS shall provide a text command line interface for directives to be entered.	Full	
UI1.3	The SIMSS shall provide the same degree of control from directives that is available from the graphical user interface.	Partial	Not all modules
UI1.4	The SIMSS shall be capable of generating a scenario file containing all the directives entered by the user during a specific period of time.		
UI1.5	The SIMSS shall provide the capability to recall previously entered directives to be executed again or edited and then executed up to at least the last ten directives entered	Full	
UI2	The SIMSS shall provide the user with project control.	Full	
UI2.1	The SIMSS shall provide the user with the capability to create and delete projects.	Full	
UI2.2	The SIMSS shall provide the user with the capability to select which server host to connect to for a specific project.	Full	
UI2.3	The SIMSS shall provide the user with the capability to add a module to a project.	Full	
UI2.4	The SIMSS shall provide the user with the capability to delete a module from a project.	Full	
UI3	The SIMSS shall provide the user with module	Full	

	channel control.		
UI3.1	The SIMSS shall provide the user with the capability to determine how many input and output channels a module is capable of handling.	Full	
UI3.2	The SIMSS shall provide the user with the capability to create a link between an output channel of any module to an input channel of any other module.	Full	
UI3.3	The SIMSS shall provide the user with the capability to delete any link previously created.	Full	
UI3.4	The SIMSS shall provide the user with the capability to determine which channels a link connects.	Full	
UI3.5	The SIMSS shall be capable of displaying meaningful names for each module input or output channel.		
UI4	The SIMSS shall be capable of displaying a brief description of a module's function.	Full	
UI5	The SIMSS shall provide the user with the capability to stop and start operations for a specific project.	Full	
UI6	The SIMSS user interface shall be capable of displaying and logging all system event messages for a specific project.	Full	
UI6.1	The SIMSS user interface shall log all system messages for a specific project to an event log that is accessible offline if SIMSS is not running.	Full	
UI6.2	The SIMSS user interface shall provide a display showing all system messages generated during the current session for a specific project.	Full	
UI7	The SIMSS user interface shall be capable of printing any display.	Full	Copy/paste to/from clipboard
UI8	The SIMSS shall provide the user with the capability to save the configuration of a current project.	Partial	
UI8.1	The SIMSS shall be capable of saving the overall configuration (modules and links) of a project.	Full	
UI8.2	The SIMSS shall be capable of saving the configuration information specific and internal to modules in a project.	Partial	Not all modules
UI9	The SIMSS shall provide the user with the capability to restore a project based on a previously stored configuration.	Partial	
UI9.1	The SIMSS shall be capable of restoring the	Full	

	overall configuration (modules and links) of a project based on a saved configuration		
UI9.2	The SIMSS shall be capable of restoring the configuration information specific and internal to modules in a project based on a saved configuration.	Partial	Not all modules
UI10	Each SIMSS module shall provide the user with the capability to control its configuration and functionality.	Partial	
UI11	Each SIMSS module shall provide the user with the capability to monitor its configuration and status.	Full	
UI12	The SIMSS user interface shall update all dynamic displays at least once every five seconds.	Full	
UI13	Each SIMSS module shall be capable of displaying to the user the date and release number of the version currently running.	Partial	It may not reflect the latest date of changes if the *DllMainClass.cpp was not checked in at the same time.

DM	***Data Management Requirements***	***R4.1 Impl.***	***Comments***
DM1	The SIMSS shall include a data format control document (DFCD) that defines a standard storage format and medium for all information needed to generate and modify telemetry, validate and identify commands, and reflect commands in telemetry.		
DM2	All SIMSS modules that generate or modify telemetry, validate or identify commands, or reflect commands in telemetry shall adhere to the DFCD when storing and retrieving data used for these purposes.		
DM3	The DFCD shall include record formats for information about each of the following telemetry elements: a. Telemetry parameters b. Telemetry locations c. Telemetry packets d. TDM telemetry formats		

	e. Physical channels f. Virtual channels g. Virtual channel to physical channel mappings h. Packet to virtual channel mappings i. Polynomial conversions between raw telemetry values and engineering units j. Linear conversions between raw telemetry values and engineering units k. Conversions between raw telemetry values and discrete state text l. Red and yellow limit values		
DM4	The DFCD shall include record formats for information about each of the following command elements: a. CCSDS commands b. Non-CCSDS commands c. Command data area parameters d. Command data area parameter conversions		
DM5	The DFCD shall include record formats for the information required to map telemetry verifiers to commands received		

DT	***Data Transport Requirements***	***R4.1 Impl.***	***Comments***
DT1	The SIMSS shall provide a module that is capable of sending data via TCP/IP and UDP/IP.	Full	
DT1.1	This module shall be capable of connecting to an IP socket for the purpose of transmitting data.	Full	
DT1.2	This module shall be capable of transmitting data over an IP socket in UDP unicast mode.	Full	
DT1.3	This module shall be capable of transmitting data over an IP socket in UDP multicast mode.	Full	
DT1.4	This module shall be capable of transmitting data over an IP socket in TCP mode.	Full	
DT1.4.1	This module shall be capable of transmitting data over an IP socket as a TCP/IP client.	Full	
DT1.4.2	This module shall be capable of transmitting data over an IP socket as a TCP/IP server.	Full	
DT1.5	This module shall be capable of transmitting up to 6000 bytes of data in a single IP data block.	Full	
DT1.6	This module shall be capable of interfacing with other modules for the purpose of accepting data to be transmitted.	Full	

DT1.7	This module shall be capable of displaying the following IP interface status information to the user: a. Number of packets transmitted b. Enabled/disabled status c. The most recent data that was transmitted	Full	
DT1.8	This module shall provide the user the capability of setting, by means of a user interface, IP socket parameters including the following: a. IP address b. Port number c. Defined or variable data size d. Multicast address	Full	
DT2	The SIMSS shall provide a module that is capable of receiving data via TCP/IP and UDP/IP.	Full	
DT2.1	This module shall be capable of connecting to an IP socket for the purpose of receiving data.	Full	
DT2.2	This module shall be capable of receiving data over an IP socket in UDP mode.	Full	
DT2.3	This module shall be capable of receiving data over an IP socket in UDP multicast mode.	Full	
DT2.4	This module shall be capable of receiving data over an IP socket in TCP mode.	Full	
DT2.4.1	This module shall be capable of receiving data over an IP socket as a TCP/IP client.	Full	
DT2.4.2	This module shall be capable of receiving data over an IP socket as a TCP/IP server.	Full	
DT2.5	This module shall be capable of receiving up to 6000 bytes of data in a single IP data block.	Full	
DT2.6	This module shall be capable of interfacing with other modules for the purpose of passing on received data.	Full	
DT2.7	This module shall be capable of displaying the following IP interface status information to the user: a. Number of packets received b. Enabled/disabled status c. The most recent data that was received	Full	
DT2.8	This module shall provide the user the capability of setting, by means of a user interface, IP socket parameters including the following: a. IP address b. Port number c. Defined or variable data size d. Multicast address	Full	

DT3	The SIMSS shall provide a module that is capable of sending serial data and clock using ISA based architecture.	Full	
DT3.1	This module shall be capable of connecting to a serial line for the purpose of transmitting data.	Full	
DT3.2	This module shall be capable of transmitting a frame length up to 4096 bytes of data.	Full	
DT3.3	This module shall be capable of interfacing with other modules for the purpose of accepting data to be transmitted.	Full	
DT3.4	This module shall be capable of selecting the internal clock in a range from 100 Hz to 4 MHz.	Full	
DT3.5	This module shall be capable of displaying the following information to the user a. Enabled/disabled status b. Data frequency c. Frame length d. Frame count e. The most recently transmitted block of data	Full	
DT3.6	This module shall provide the user the capability of setting serial interface board parameters through a user interface including the following encoding choices: a. Non-Return-to-Zero-Level (NRZ-L) (true or inverted) b. NRZ-Mark (NRZ-M) (true or inverted) c. NRZ-Space (NRZ-S) (true or inverted) d. Bi-phase-Level (BI0-L) e. BI0-M f. BI0-S	Full	
DT3.7	This module shall be capable of providing an external clock interface in the RS422/TTL standard.	Full	
DT4	The SIMSS shall provide a module that is capable of receiving serial data and clock (RS422/TTL), using ISA based architecture.	Full	
DT4.1	This module shall be capable of connecting to a serial line for the purpose of receiving data.	Full	
DT4.2	This module shall be capable of receiving a frame length of up to 4096 bytes of data.	Full	
DT4.3	This module shall be capable of interfacing with other modules for the purpose of passing on received data.	Full	
DT4.4	This module shall be capable of displaying the following information to the user:	Full	

	a. Enabled/disabled status b. Data frequency c. Frame length d. Frame count e. Subframe count f. Frame sync drop count g. Subframe drop count or sequence drop h. Frame sync search and lock status i. Buffer position – IP to serial conversion j. The most recently received block of data		
DT4.5	This module shall provide the user the capability of setting serial interface board parameters through a user interface including the following choices: a. NRZ-L (true or inverted) b. NRZ-M (true or inverted) c. NRZ-S (true or inverted) d. BI0-L (true or inverted) e. BI0-M (true or inverted) f. BI0-S (true or inverted)	Full	
DT4.6	This module shall provide the user the following information: a. Sync status (Lock, Search, Idle) b. Subframe status (Lock, Search, Idle) c. Clock status (active or inactive) d. Frame count e. Frame drop count f. Subframe count g. Subframe drop count h. Data orientation (most significant bit or least significant bit is transmitted first) i. Data polarity (true or inverted) j. Clock polarity (true or inverted) k. Auto polarity check l. Frame length m. RS422/TTL n. Correlation o. Subframe size p. Subframe location q. Subframe start count r. Subframe stop count s. Sync size (up to 8 bytes) t. Sync mask u. Sync pattern v. Fixed/Variable clock (Output only – the IP to	Full	

	serial conversion scheme. By selecting variable clock rate changes according to IP rate)		
DT4.7	This module shall provide the capability of operating in asynchronous mode	Full	
DT5	The SIMSS shall provide a module that is capable of sending serial data and clock on with PCI based serial cards.	Partial	
DT5.1	This module shall be capable of connecting to a serial line for the purpose of transmitting data.	Full	
DT5.2	This module shall be capable of transmitting up to 4096 bytes of data in a single operation.	Full	
DT5.3	This module shall be capable of interfacing with other modules for the purpose of accepting data to be transmitted.	Partial	See limitations (Attachment H of R4.1 delivery package)
DT5.4	This module shall be capable of selecting the internal clock in a range from 900 Hz to 4 MHz. (Output internal clock only. Input clock is driven by the external clock).	Partial	See limitations (Attachment H of R4.1 delivery package)
DT5.5	This module shall be capable of selecting the channel to transmit data.	Full	
DT5.6	This module shall be capable of selecting the polarity of the data stream (true or inverted).	Full	
DT5.7	This module shall be capable of selecting data orientation (most significant bit or least significant bit is transmitted first).	Full	
DT5.8	This module shall provide the user the capability of setting serial interface board parameters through a user interface including the following Pulse Code Modulation (PCM) code choices: a. NRZ-L (true or inverted) b. NRZ-M (true or inverted) c. NRZ-S (true or inverted) d. BI0-L (true or inverted) e. BI0-M (true or inverted) f. BI0-S (true or inverted)	Full	
DT5.9	This module shall be capable of selecting data encoding (CRC, Reed-Solomon, Randomization, Convolution)	Full	
DT5.10	This module shall be capable of displaying the	Full	

	following information to the user a. PCM code selection b. Data Orientation c. Channel d. Clock Type e. Data frequency f. Data polarity g. Frame length h. Frame count i. Data encoding selection j. Data patterns within a frame k. The most recently transmitted block of data		
DT5.11	This module shall be capable of providing an external clock interface in the RS422/TTL standard.		
DT6	The SIMSS shall provide a module that is capable of receiving serial data and clock (RS422/TTL) with PCI based serial cards.	Full	
DT6.1	This module shall be capable of connecting to a serial line for the purpose of receiving data.	Full	
DT6.2	This module shall be capable of receiving up to 4096 bytes of data in a single operation.	Full	
DT6.3	This module shall be capable of interfacing with other modules for the purpose of passing on received data.	Full	
DT6.4	This module shall be capable of providing an external clock interface in the RS422 standard.	Full	
DT6.5	This module shall be capable of selecting data orientation (most significant bit or least significant bit is received first).	Full	
DT6.6	This module shall be capable of selecting data polarity (true, inverted, or auto polarity check).	Full	
DT6.7	This module shall be capable of selecting synchronized patterns up to 4 bytes.	Full	
DT6.8	This module shall be capable of selecting FIFO size up to 99 buffers	Full	
DT6.9	This module shall be capable of selecting data input type (PDP and SIM).	Full	
DT6.10	This module shall be capable of selecting of setup command information (Tail sequence, tail length, tail pattern, and max command size).	Full	
DT6.11	This module shall be capable of setting up maximum value, size in bits, and starting location in bits, of sub frame.	Full	

DT6.12	This module shall provide the user the capability of setting serial interface board parameters through a user interface including the following choices: a. NRZ-L (true or inverted) b. NRZ-M (true or inverted) c. NRZ-S (true or inverted)	Full	
DT6.13	This module shall be capable of displaying the following information to the user: a. Number of frames received b. Number of frames dropped c. Number of subframe received d. Number of subframe dropped e. PCM coding f. Data polarity g. Data orientation h. Receiving channel i. Data input type j. Sync. Pattern k. Sync. Size l. Frame size m. FIFO size n. Tail length o. Tail pattern p. Tail sequence enabled/disabled status q. Maximum command size r. Maximum value of subframe counter s. Minimum value of subframe counter t. Subframe counter size in bits u. Subframe starting location in bits v. Status of frame synchronization and subframe sequence w. Data pattern within one frame x. The most recently received block of data	Full	
DT7	The SIMSS shall be able to operate in IP and serial modes simultaneously.	Full	
DT8	The SIMSS shall support IP to Serial and Serial to IP data conversion and buffering.	Partial	Avtec cards only
DT9	The SIMSS shall provide a module that is capable of using two Ethernet cards simultaneously.	Full	
DT9.1	This module shall be able to select which Ethernet card is default card	Full	
DT9.2	This module and the Ethernet card it is using shall be able to communicate independently of any other Ethernet card in the system. This includes	Full	

	requirements DT1.1 – DT1.8, DT2.1-DT2.8.		
DT9.3	This module shall be able to connect to any Ethernet, independent of any other card.	Full	
DT10	The SIMSS shall provide a module that is capable of formatting data into message blocks with standard or user-defined formats.	Partial	
DT10.1	This module shall be capable of connecting to other modules for the purpose of receiving data to be formatted.	Full	
DT10.2	This module shall be capable of connecting to other modules for the purpose of passing along formatted data.	Full	
DT10.3	This module shall be capable of creating messages with the following formats (this implies the capability to block data into these formats): a. NASA Communications (NASCOM) Johnson Space Center (JSC) blocks b. NASCOM Multiplexer-Demultiplexer (MDM) blocks c. NASCOM Deep Space Network (DSN) block d. NASCOM DSN/GSFC Interface Blocks (DGIB) e. NASCOM JSC to Payload Operations Control Center (POCC) Blocks f. NASCOM JSC to Ground Space Tracking Data Network (GSTDN) Blocks g. Real-Time Protocol (RTP) messages h. EOS Data Operations System (EDOS) ground message header messages i. EDOS service header messages	Partial	a, b, c, d, e, f, g only; a includes POCC to JSC command blocks.
DT10.4	This module shall be capable of adding message headers and trailers based on the contents of a text file providing formatting information according to SIMSS specifications.	Full	Includes CCS T3ICD format
DT10.5	This module shall provide a user interface that allows the user to set the following parameters: a. The type of message to be formatted and configurable parameters associated with that type b. The pathname of the file to be used as the basis of formatting	Partial	b only
DT10.6	This module shall provide a status display that shows the user the following information: a. The number of data packets received	Partial	a, b only

	b. The number of data blocks sent c. The current type of message being formatted d. Configuration parameters e. Last block transmitted		
DT10.7	This module shall provide the capability to transmit Circuit Assurance Blocks (CABs). The CAB blocks will be transmitted at a rate of 1 block per five seconds when no telemetry blocks are active.		
DT10.8	This module shall provide the capability to transmit 'empty' NASCOM blocks in the configured block type. The 'empty' blocks shall be defined as NASCOM blocks with a data length and rate consistent with the configured telemetry rate where the data content is fill data.		
DT11	The SIMSS shall provide a module that is capable of validating message blocks with standard or user-defined formats and extracting and sending the enclosed data on to another SIMSS module.	Partial	
DT11.1	This module shall be capable of connecting to other modules for the purpose of receiving message data to be validated.	Full	
DT11.2	This module shall be capable of connecting to other modules for the purpose of sending message data that has been extracted from data received.	Full	
DT11.3	This module shall be capable of validating messages with the following formats: a. NASA Communications (NASCOM) Johnson Space Center (JSC) blocks b. NASCOM Multiplexer-Demultiplexer (MDM) blocks c. NASCOM Deep Space Network (DSN) blocks d. NASCOM DSN/GSFC Interface Blocks (DGIB) e. NASCOM JSC to Payload Operations Control Center (POCC) Blocks f. NASCOM JSC to Ground Space Tracking Data Network (GSTDN) Blocks g. Real-Time Protocol (RTP) messages h. EOS Data Operations System (EDOS) ground message header messages i. EDOS service header messages	Partial	a, b, c, d, e, f, g only; a includes POCC to JSC command blocks.
DT11.4	This module shall be capable of validating	Full	Includes

	message headers and trailers based on the contents of a text file providing formatting information according to SIMSS specifications.		CCS T3ICD format
DT11.5	This module shall provide a user interface that allows the user to set the following parameters: a. The type of message to be formatted and configurable parameters associated with that type b. The pathname of the file to be used as the basis of formatting	Partial	Pathname only
DT11.6	This module shall provide a status display that shows the user: a. The number of data blocks received b. The number of data packets sent c. The current type of message being validated d. Configuration parameters e. Last block received	Partial	a, b only
DT11.7	This module shall provide the capability to monitor the reception of CABs on selected input channels.		
DT11.8	This module shall provide the capability to monitor the reception of 'empty' blocks on selected input channels.		
DT12	The SIMSS shall provide a module that is capable of encoding serial data via software and of transmitting the resulting encoded data.	Full	
DT12.1	This module shall be capable of adding 16-bit cyclic redundancy check (CRC-16) encoding to a serial data stream.	Full	
DT12.2	This module shall be capable of adding Reed-Solomon check symbols to a serial data stream.	Full	
DT12.3	This module shall be capable of adding convolutional encoding to a serial data stream.	Full	
DT12.4	This module shall be capable of adding randomization encoding to a serial data stream.	Full	
DT12.5	This module shall allow the user to specify the transmission encoding methods to be performed on a serial data stream.	Full	
DT12.6	This module shall allow the user to specify the data block size to be encoded.	Full	
DT12.7	This module shall be capable of displaying the status and configuration information to the user, including: a. The current encoding methods being performed on the data stream	Full	

	b. The contents of the last data block encoded c. The total number of data blocks transmitted		
DT13	The SIMSS shall provide a module that is capable of supporting command echo.	Partial	
DT13.1	This module shall be capable of connecting to another module for the purpose of receiving data to be echoed.	Full	
DT13.2	This module shall be capable of connecting to another module for the purpose of sending data that has been echoed.	Full	
DT13.3	This module shall be capable of extracting the source and destination identifiers from a data block received, swapping them, re-generating the CRC for the block if applicable, and sending out the resulting block.	Full	
DT13.4	This module shall provide the user with the capability to set the following parameters: a. Location of the source identifier in the block b. Size of the source identifier c. Location of the destination identifier in the block d. Size of the destination identifier	Partial	Settable from a configura-tion file
DT13.5	This module shall provide the following status information to the user: a. Number of blocks received b. Current location being used for the source identifier c. Current location being used for the destination identifier d. Last source and destination identifiers received e. Contents of the most recent block received	Partial	a and e
DT14	The SIMSS shall support continuous, intermittent, and discrete transmission modes.		

TG	***Telemetry Generation Requirements***	***R4.1 Impl.***	***Comments***
TG1	The SIMSS shall provide a module that generates CCSDS telemetry based on information stored in a standard database.	Partial	
TG1.1	This module shall generate telemetry packets according to CCSDS standards.	Partial	AOS only
TG1.1.1	This module shall provide a standard packet	Full	

	primary header with an incrementing packet counter.		
TG1.1.2	This module shall provide the database-driven option of a secondary header in the one of the following formats: a. SMEX secondary header b. EOS secondary header		
TG1.1.3	This module shall use the database to define the following packet-level parameters: a. Application id b. Secondary id c. Source d. Existence of a secondary header e. Type of the secondary header f. Flag to indicate if packet is sent out on a timed basis g. If timed, time from start that packet is first sent out h. If timed, time interval between sending out packets i. Length of packet		
TG1.1.4	This module shall provide the user with the capability to: a. Set any value in a packet primary header b. Set any value in a packet secondary header c. Set any value in the data area of a packet d. Set a pattern in the data area of a packet e. Enable or disable the sending of a packet on a timed basis f. Change the interval between sending out packets	Partial	All except b,c,d.
TG1.1.5	This module shall be capable of displaying to the user: a. A packet's primary header b. A packet's secondary header c. A packet's data area d. Whether a packet is currently being sent out on a timed basis e. The interval at which a packet is being sent out		
TG1.2	This module shall be capable of packing packets into transfer frames or VCDUs and sending them on a virtual channel according to CCSDS or CCSDS AOS standards.	Partial	AOS only

TG1.2.1	This module shall be capable of either splitting or not splitting packets across frames based on parameters in the database.	Partial	User input, not database
TG1.2.2	This module shall generate a frame header with an incrementing frame count.	Full	
TG1.2.3	This module shall use the database to define the following transfer frame and virtual channel parameters: a. Virtual channel identifier b. Frame size c. Packet splitting flag d. Standard or AOS flag e. Mapping of packets to virtual channels	Partial	Not c,d; remaining parameters from flat file only
TG1.2.4	This module shall provide the user with the capability to: a. Set any value in the frame header for a virtual channel, either once or constantly until disabled b. Set any value in the data area of a frame for a virtual channel, either once or constantly until disabled c. Change the mapping of packets to virtual channels d. Enable or disable a virtual channel		
TG1.2.5	This module shall be capable of displaying to the user: a. The most recent frame header for a virtual channel b. The most recent frame contents for a virtual channel c. The packet mapping for a virtual channel d. The number of packets received for a virtual channel e. The number of frames sent on a virtual channel		
TG1.3	This module shall be capable of combining the frames for a virtual channel into a physical channel according to CCSDS standards.	Full	
TG1.3.1	This module shall be capable of sending telemetry data over one to three physical channels, each of which shall correspond to a SIMSS channel or link.	Full	
TG1.3.2	This module shall use the database to define the following physical channel parameters: a. Number of physical channels b. Virtual channel to physical channel mapping	Partial	a-c from flat file

	c. Virtual channel priority on the physical channel d. Whether the frame for the virtual channel must be full to be transmitted		
TG1.3.3	This module shall provide the user with the capability to: a. Enable or disable a physical channel b. Change the virtual channel to physical channel mapping c. Change the priority of a virtual channel on a physical channel d. Change the must-be-full flag for a virtual channel	Partial	a only
TG1.3.4	This module shall be capable of displaying to the user: a. The current number of physical channels b. Whether each channel is enabled or disabled c. The virtual channels mapped to a physical channel d. The priority of a virtual channel on a physical channel e. The must-be-full flag for a virtual channel f. The number of frames sent over a physical channel g. The most recent frame sent over a physical channel	Partial	b and f only
TG1.4	This module shall be capable of generating fill packets and frames as necessary according to CCSDS standards and database parameters. Virtual channel 7 (for standard CCSDS) and virtual channel 63 (for CCSDS AOS) shall be channels consisting entirely of fill frames.		
TG1.5	This module shall be capable of handshaking with the serial or another external module for the purpose of sending out data at a rate defined at that module.	Full	
TG2	The SIMSS shall provide a module that generates time-division multiplexed (TDM) telemetry based on information by the user.	Partial	
TG2.1	This module shall generate multiple telemetry formats on a minor frame/major frame basis.	Full	One format at a time
TG2.2	This module shall be capable of sending telemetry over 1 channel.	Full	
TG2.3	This module shall use a configuration file or GUI to define the following parameters:	Full	a-d settable by user.

	a. Number of minor frames per major frame b. Size of a minor frame c. Position of the minor frame counter in the minor frame d. Position of the major frame counter in the minor or major frame		
TG2.4	This module shall be capable of adding valid CRC check words to the end of a minor frame.	Full	16 or 32 bits
TG2.5	This module shall be capable of accepting the value of a any parameter (e.g., a command counter) from an external module.	Full	Currently on byte boundary
TG2.6	This module shall provide the user with the capability to: a. Enable or disable the physical channel b. Set any value in a minor frame c. Set patterns in a minor frame or sequence of minor frames, either consecutive or subcommutated d. Enable or disable setting the CRC check words	Partial	a-d (no subcoms)
TG2.7	This module shall be capable of displaying to the user: a. Whether each channel is enabled or disabled b. The contents of the most recent minor frame to be sent from that physical channel c. The minor and major frame counts for a physical channel d. The total number of frames sent out over the physical channel	Partial	a, c, d only. b use Monitor or Test-Module to view.
TG2.8	This module shall be capable of handshaking with the serial or another external module for the purpose of sending out data at a rate defined at that module.	Full	
TG3	The SIMSS shall provide a generic, data-driven module that is capable modifying TDM telemetry.	Full	
TG3.1	This module shall be capable of ingesting a data stream containing only TDM minor frames, perform modifications on the data, and output the modified data.	Full	
TG3.2	This module shall provide the user with the capability to define the: a. Minor frame size in bytes.	Full	

	b. Minor frame counter size in bits. c. Minor frame counter location at the bit level		
TG3.3	This module shall be capable of providing mod-bit functionality.	Full	
TG3.3.1	This module shall be capable of displaying to the user: a. The current minor frame number. b. The data value in the telemetry stream before modification. c. The data value in the telemetry stream after modification.	Full	
TG3.3.2	This module shall provide the user with the capability to: a. Enable or disable any modification. b. Define the start frame. c. Define the subcom depth. d. Define the start byte. e. Define the start bit. f. Define the bit length. g. Define the value to be placed in the minor frame.	Full	
TG3.4	This module shall be capable of modifying telemetry via the use of defined telemetry mnemonics.		
TG3.4.1	This module shall be capable of receiving and implementing telemetry mnemonic updates provided by a modeling interface module.		
TG3.4.2	This module shall be capable of ingesting an ascii text file (format TBD) that contains a list of telemetry mnemonics and their telemetry locations.		
TG3.4.3	This module shall provide the user the capability to set individual telemetry mnemonics to PCM (or raw) values.		
TG3.5	This module shall be capable of receiving and implementing telemetry mnemonic updates provided by a command ingest module.		
TG3.6	This module shall be fully controllable from the scenario module.		

CI	***Command Ingest Requirements***	***R4.1 Impl.***	***Comments***
CI1	The SIMSS shall provide a module with the capability to receive, validate, and identify CCSDS commands.	Partial	
CI1.1	This module shall be capable of receiving commands	Full	

	in the form of CLTUs.		
CI1.2	This module shall be capable of performing the following CCSDS validation checks: a. CLTU header and trailer b. Codeblock CRC c. Transfer frame header d. Frame Acceptance and Reporting Mechanism e. Packet primary header	Full	
CI1.3	This module shall generate a command link control word (CLCW) for each virtual channel that reflects the commands received.	Full	
CI1.4	This module shall be capable of receiving and executing CCSDS FARM special commands, including: a. Unlock FARM b. Set expected frame sequence number	Full	
CI1.5	This module shall be capable of using a database to determine if a command received is valid.		
CI1.6	This module shall provide the user with the capability to enable or disable any validation check.	Full	
CI1.7	This module shall provide the user with the capability to set or change the following parameters: a. Expected CLTU header b. Expected CLTU trailer c. Codeblock size d. Spacecraft id (SCID) e. Any field in the CLCWs, including 1. Virtual channel id (VCID) 2. Next expected frame sequence number f. FARM sliding window size g. Database source and version to use for validation and identification	Partial	g. not implemented
CI1.8	This module shall be capable of displaying to the user: a. The most recent CLTU received b. The codeblock data areas from the most recent CLTU received c. The most recent transfer frame received for each virtual channel d. The most recent packet received for each virtual channel e. The current CLCW for each virtual channel f. Counts of valid and invalid command elements received	Partial	d. has a problem
CI1.9	This module shall be capable of generating event	Full	

	messages when errors are seen that provide specifics on the error.		
CI1.10	This module shall be capable of generating an event message when a valid command is received that provides the command mnemonic and information on its contents.		
CI1.11	This module shall provide the user with the capability to suspend/resume the update of CLCW upon completion of command validation.	Full	
CI1.12	This module shall be capable of providing the current CLCW for any virtual channel to another module.	Full	
CI2	The SIMSS shall provide a module with the capability to receive, validate, and identify non-CCSDS commands.	Partial	
CI2.1	This module shall be capable of receiving a stream of non-CCSDS commands.	Full	
CI2.2	This module shall be capable of performing the following validation checks on non-CCSDS commands: a. Spacecraft identifier b. Hamming code c. Command block format including barker code, preamble, and postamble	Full	
CI2.3	This module shall be capable of using a database to determine if a command received is valid.		
CI2.4	This module shall provide the user with the capability to enable or disable any validation check.	Full	
CI2.5	This module shall provide the user with the capability to set or change the following parameters: a. Expected spacecraft identifier b. Expected barker code c. Expected pre-amble and post-amble pattern d. Expected pre-amble and post-amble length e. Internal command counter f. Database source and version to use for validation and identification g. Location(s) of command counter in telemetry	Partial	a-e and g only
CI2.6	This module shall be capable of displaying to the user: a. The most recent command received b. Counts of valid and invalid command elements received	Full	
CI2.7	This module shall be capable of generating event messages when errors are seen that provide specifics on the error.	Full	

CI2.8	This module shall be capable of generating an event message when a valid command is received that provides the command mnemonic and information on its contents.		
CI2.9	This module shall provide the user with the capability to suspend/resume the update of command counter(s) upon completion of command validation.	Full	
CI2.10	This module shall be capable of providing the current command counter(s) to another module.	Full	

CG	***Command Generation Requirements***	***R4.1 Impl.***	***Comments***
CG1	The SIMSS shall provide a module that is capable of creating, saving, reading, modifying, and transmitting telecommand headers and binary files under user control.	Partial	
CG1.1	The module shall support the following predefined types of CCSDS data buffers: a. Transfer frame header b. Packet primary header c. Packet secondary header d. Command data e. Command	Partial	
CG1.1.1	The module shall allow the user to construct a data buffer of a predefined type (CG1.1a-e) and save it as a binary data file.	Partial	
CG1.1.2	The module shall allow the user to identify a binary data file as a predefined type (CG1.1a-e) and will interpret the file accordingly.		
CG1.1.3	The module shall allow the user to identify a portion of a binary data file by the start byte and number of bytes as a predefined type (CG1.1a-e) and will interpret the portion of the file accordingly.	Partial	
CG1.1.4	The module shall allow the user to change any field in a predefined type (CG1.1a-e) buffer.	Full	
CG1.1.5	The module shall automatically increment counter fields in a predefined type (CG1.1a-e) buffer unless overridden by the user.	Full	
CG1.2	The module shall be capable of generating CCSDS composite files.	Full	
CG1.2.1	The module shall allow the user to combine separate files (from the predefined list of	Full	

	data type CG1.1a-e) into a single file. (For example, a command may consist of a transfer frame header, a packet primary header, a packet secondary header, and command data.)		
CG1.2.2	The module shall be capable of displaying to the user the individual components of a composite file and the identity information for each of those components.		
CG1.2.3	The module shall maintain identify information about any binary data file that contains other than raw, noncomposite data. This identify information shall be kept separate from the data file and shall indicate what the file represents if other than raw data.	Full	
CG1.3	The module shall be capable of processing raw data files containing CCSDS-formatted command data.		
CG1.3.1	The module shall be capable of converting a raw data file into codeblocks according to CCSDS specifications.		
CG1.3.2	The module shall be capable of calculating CRC and any polynomial check defined in the CCSDS specification for a raw data file.		
CG1.3.3	The module shall be capable of calculating CRC and any polynomial check defined in the CCSDS specification for data entered by the user.		
CG1.4	The module shall be capable of processing raw data files containing non-CCSDS formatted command data.		Will be implemented in TDM Cmd Gen Module
CG1.4.1	The module shall be capable of adding a barker code and a hamming code to command data from a raw data file.		Will be implemented in TDM Cmd Gen Module
CG1.4.2	The module shall be capable of converting command data from a raw data file between NRZ-L and NRZ-M data formats.		Will be implemented in TDM Cmd Gen Module
CG1.5	The module shall be capable of transmitting CCSDS and non-CCSDS commands in real-time.	Partial	
CG1.5.1	The module shall be capable of sending the		

	contents of a file as selected by the user.		
CG1.5.2	The module shall provide the user with the capability to send data once.		
CG1.5.2	The module shall provide the user with the capability to send data for a fixed number of times at a user-defined interval.		
CG1.5.3	The module shall provide the user with the capability to send data at a user-defined interval until manually stopped.		
CG1.5.4	The module shall provide the same capabilities available for binary file generation and manipulation to an internal data buffer.		
CG1.6	The module shall be capable of generating and saving spacecraft/user profiles.		
CG1.6.1	The module shall be capable of generating a spacecraft/user profile that will contain spacecraft-specific CCSDS information including the following: a. Codeblock size b. Spacecraft id		
CG1.6.2	The module shall be capable of generating a spacecraft/user profile that will contain spacecraft-specific non-CCSDS information including the following: a. Command length b. Spacecraft id c. Barker code d. Location of hamming code e. Input code (NRZ-L, NRZ-M) f. Preamble and postamble		Will be implemented in TDM Cmd Gen Module.
CG1.6.3	The module shall provide the user with the capability to create, edit, and save a profile.		
CG1.6.4	The module shall allow the user to specify the profile to use for current operations.		
CG1.6.5	The module shall not limit the number of profiles that the user can create or use.		
CG1.7	The module shall provide the user with the capability of typing in a command mnemonic and submnemonics (if appropriate) and constructing a command therefrom by interfacing with the SIMSS commanding operations database (ODB).		

DA	*Data Analysis Requirements*	*R4.1 Impl.*	*Comments*
DA1	The SIMSS shall provide a generic, data-driven module that is capable of CCSDS telemetry VC quality monitoring and decommutation. (VC Processor)	Full	
DA1.1	This module shall be capable of monitoring a telemetry stream in CCSDS or CCSDS-AOS format.	Full	CCSDS-AOS only
DA1.2	This module shall be capable of transmitting user selected VCs.	Full	
DA1.3	This module shall be capable of validating transfer frames or VCDUs, including CRC check, RS check, and (De)Randomizaion. a. CRC error count b. RS error count c. RS uncorrectable error count d. Randomization (not displayed) e. VC sequence error	Full	VCDUs only
DA1.4	This module shall be capable of validating the following fields in a transfer frame or VCDU header: a. Version b. Spacecraft ID c. Virtual Channel ID d. VCDU counter e. Replay flag f. Spare bits		Validation of these parameters is not implemented in R4.1.
DA1.5	This module shall be capable of displaying to the user: a. The most recent transfer frame or VCDU received b. The parsed header of the most recent transfer frame or VCDU received c. The number of transfer frames or VCDUs received with and without errors d. VCDU sequence error count e. The number of transfer frames or VCDUs received for each virtual channel. f. CRC error count g. RS error count h. RS uncorrectable error count	Partial	VCDU display only. h. RS uncorrectable error is not included in R4.1.
DA2	a. This module shall be capable of extracting packets from the VCDU and	Full	

	display packet information on status screen. (Packet Processor)		
DA2.1	This module shall be capable of displaying to the user: a. The most recent packet received with a given APID b. The parsed header of the most recent packet received with a given APID c. The number of packets received with a specific APID with and without errors d. Time interval between specific APIDs	Full	Packet displays will not be in this module, rather the Monitor module will be used to display the selected APID packets.
DA2.2	This module shall provide the user with the capability to: a. Define whether to expect packets b. Define the valid packet APIDs c. Define expected content values in specific packets by APID	Partial	Packet control only.
DA2.3	This module shall be capable of using the database for the following information: a. Whether the packets in a frame should be split between frames b. Valid packet APIDs c. Valid packet lengths d. Valid packet header values e. The locations of telemetry parameters within a packet	Partial	The function of a-e are implemented but not using database.
DA2.4	This module shall have option to select which APIDs can be forwarded to the next module.	Partial	APID packets filtered.
DA3	The SIMSS shall provide a generic, data-driven module that is capable of TDM telemetry data quality monitoring.	Partial	
DA3.1	This module shall be of capable of validating the following fields in a TDM telemetry stream: a. Sync pattern b. Minor frame count c. Major frame count d. User-defined parameters	Full	
DA3.2	This module shall be capable of displaying to the user the following rate-dependent items: a. The most recent minor frame received b. The most recent minor frame counter seen c. The most recent major frame counter seen d. The number of minor frames received, with	Partial	e is not implemented.

	and without errors e. The number of major frames received, with and without errors f. Telemetry data, in raw format, extracted from the stream based on user information about size and position g. The current value of the command counter(s)		
DA3.3	This module shall provide the user with the capability to: a. Enable or disable any element of the validation process b. Define the size of a minor frame c. Define the number of minor frames in a major frame using minimum and maximum value d. Define the size, value, and position of the expected sync pattern e. Define the size and position of the minor frame counter f. Define the size and position of the major frame counter g. Define the size, position, and subcommutation of parameters to display or validate h. Save and restore the user-defined configuration	Full	
DA3.4	This module shall be capable of extracting and forwarding data to another module, via no less than 3 output channels, defined in the following manner: a. Minor frames – all, one individual frame, or using subcom depth b. User-defined areas within minor frames by start byte and number of bytes	Full	
DA3.5	This module shall provide the capability to ingest asynchronous normal and inverted data and sync align data using a maximum sync pattern of 32 bits.		

DA4	The SIMSS shall provide a module with a capability of displaying a data stream in operator selectable display formats.	Full	
DA4.1	This module shall be capable of displaying data	Full	

	in decimal.		
DA4.2	This module shall be capable of displaying data in 8, 16, or 32 bit hexadecimal.	Full	
DA4.3	This module shall be capable of displaying data in 8, 16, or 32 bit octal.	Full	
DA4.4	This module shall be capable of shifting the displayed data 1- 31 bits, left or right.	Full	
DA4.5	This module shall be capable of inverting the displayed data.	Full	
DA4.6	This module shall be capable of converting the displayed data to/from NRZ-L and NRZ-M formats.	Full	
DA5	The SIMSS shall provide a module with a capability of encoding data from a data stream.	Partial	
DA5.1	This module shall be capable of adding CRC encoding to a data stream.	Full	
DA5.2	This module shall be capable of adding Reed-Solomon check symbols to a data stream.	Full	
DA5.3	This module shall be capable of adding convolutional encoding to a data stream.	Full	
DA5.4	This module shall be capable of adding randomization encoding to a data stream.	Full	
DA5.5	This module shall be capable of de-randomizing data from a data stream.	Partial	Randomization is also de-randomization. Instead of creating additional decoder module, this shall be used as de-randomizer as well.
DA5.6	This module shall provide the user with the capability to specify the encoding to be performed on a data stream.	Full	
DA5.7	This module shall be capable of displaying status and configuration information to the user, including: a. Frame Count b. CRC-16 enabled/disabled c. Convolution enabled/disabled d. Randomization enabled/disabled e. Reed-Solomon enabled/disabled f. Interleave g. Virtual Fill	Full	

	h. Frame Size in bytes i. Sync Pattern Size in bytes		

TM	*Timing Requirements*	*R4.1 Impl.*	*Comments*
TM1	The SIMSS shall be capable of using a timing card to drive interrupts and system time.		
TM2	The SIMSS shall be capable of generating time formats.	Partial	
TM2.1	The SIMSS shall be capable of generating GMT.	Full	
TM2.2	The SIMSS shall be capable of generating UTC.		
TM2.3	The SIMSS shall be capable of generating UT1.		
TM3	The SIMSS shall be capable of formatting time data from one of the standard time formats into any of the following NASA standard time formats: a. PB4 b. PB5 c. Small Explorer (SMEX) packet header time d. CCSDS unsegmented time code	Partial	a, c, d only
TM4	The SIMSS shall be capable of generating mission specific time formats created by plug-in modules based on standard formats.		
TM5	The SIMSS shall control timed activities in a simulated accelerated mode.		

DR	*Data Archiving Requirements*	*R4.1 Impl.*	*Comments*
DR1	The SIMSS shall provide a module capable of storing the contents of a data stream to disk files (i.e. logging).	Full	
DR1.1	This module shall be capable of opening a disk file upon request from other modules.	Full	
DR1.2	This module shall be capable of closing a disk file upon request from other modules.	Full	
DR1.3	This module shall be capable of appending 6000 bytes of data to an open disk file in one operation.	Full	
DR1.4	This module shall be capable of interfacing with other modules for the purpose of transferring data from those modules.	Full	
DR1.5	This module will provide a user interface for	Full	

	the purpose of reporting status information including the following: a. Enabled or disabled status of logging b. Number of bytes written to a log file		
DR1.6	This module will provide a user interface for the purpose of entering the following information: a. The maximum size of a log file b. The name of a disk file	Full	
DR2	The SIMSS shall provide a module that is capable of reading the contents of a disk file and sending it out as a data stream.	Partial	
DR2.1	This module shall be capable of opening a disk file upon request from another module.		
DR2.2	This module shall be capable of closing a disk file upon request from another module.		
DR2.3	This module shall be capable of reading 6000 bytes of data from an open disk file in one operation.	Full	
DR2.4	This module shall provide the user with the capability to set the following parameters: a. Pathname of the file to read from the disk b. Size (in bytes) of a block of data to read from the disk and send out at one time c. Offset (in bytes) from the beginning of the file where to start reading and sending data d. Output mode, including manual mode as described in DR2.6 and automatic modes as described in DR2.7 e. File read mode as described in DR2.8	Partial	e is not complete.
DR2.5	This module shall provide the following display and status information to the user: a. The pathname of the file being transmitted b. The number of blocks transmitted from the file c. The current position in the file d. The size of the file	Full	
DR2.6	This module shall provide a manual output mode where each block of data is loaded from	Full	

	the disk and sent individually under user control.		
DR2.7	This module shall provide automatic output modes that include the capability to: a. Send out the contents of a file once, several times, or continuously b. Send out a subset of a file once, several times, or continuously c. Send out the blocks in a file or subset of a file one or more times before sending out the next block	Full	
DR2.8	This module shall provide file read modes that include the capability to: a. Load consecutive blocks from a file based on a fixed offset b. Load consecutive blocks from a file based on a synchronization pattern at the beginning of each block c. Load consecutive blocks from a file based on a length field within each block d. Load consecutive blocks from a file based on a header added by the log module	Partial	a, d only
DR3	The SIMSS shall store and retrieve data to and from disk files on various storage devices.	Full	
DR3.1	The SIMSS shall store and retrieve data to and from hard drives.	Full	
DR3.2	The SIMSS shall store and retrieve data to and from CDs.	Full	
DR3.3	The SIMSS shall store and retrieve data to and from Zip drives.	Full	
DR4	The SIMSS shall store and retrieve data via in-line FTP.		

MD	***Modeling Requirements***	***R4.1 Impl.***	***Comments***
MD1	The SIMSS shall supply an interface to allow remote manipulation of internal data points.	Partial	SIMSS can now receive (name, value) data from Model Generator Prototype.
MD2	The SIMSS shall provide the capability to model internal and telemetry parameters based on orbital position.		

MD2.1	This capability shall allow the user to set the following orbit parameters: a. Time of orbit start b. Orbit period c. Eclipse duration time per orbit d. Time from orbit start until eclipse		
MD2.2	This capability shall support the following modeling types: a. Sine wave b. Ramping c. Exponential d. Natural log e. Polynomials up to the fifth order f. Table-driven with interpolation g. Table-driven without interpolation		
MD2.3	This capability shall be capable of using either raw or engineering units when modeling.		
MD2.4	This capability shall be capable of using the database to define: a. Specific models (model type plus type-specific parameters) b. Associations between models, parameters, and orbit status (day/night) c. Granularity (how often to update based on model) in seconds		
MD2.5	This capability shall allow the user to see, enable or disable, or change any model read from the database.		
MD2.6	This capability shall allow the user to define additional models and associations.		
MD2.7	This capability shall be capable of displaying: a. The current and next value of any modeled parameter (EU or raw) b. Time until next value is applied c. The model being used for a modeled parameter d. A graphic plot of recent values for any modeled parameter		
MD3	The SIMSS shall provide the capability to model internal and telemetry parameters based on spacecraft events.		
MD3.1	Spacecraft events shall include: a. Specific command received b. Operator directive indicating that an		

	event has occurred c. Telemetry or internal parameter going into a specific range d. Telemetry or internal parameter going out of a specific range e. Reaching a specific spacecraft time		
MD3.2	This capability shall support the following modeling types: a. Sine wave b. Ramping c. Exponential d. Natural log e. Polynomials up to the fifth order f. Table-driven with interpolation g. Table-driven without interpolation		
MD3.3	This capability shall be capable of using either raw or engineering units when modeling.		
MD3.4	This capability shall be capable of using the database to define: a. Specific models (model type plus type-specific parameters) b. Associations between models, parameters, and events c. Granularity (how often to update based on model) in seconds		
MD3.5	This capability shall allow the user to see, enable or disable, or change any model read from the database.		
MD3.6	This capability shall allow the user to define additional models and associations.		
MD3.7	This capability shall be capable of displaying: a. The current and next value of any modeled parameter (EU or raw) b. Time until next value is applied c. The model being used for a modeled parameter d. A graphic plot of recent values for any modeled parameter		
MD4	The SIMSS shall supply an interface to allow user or mission-specific extensions to model science instrument data.		
MD5	The SIMSS shall provide a module with the capability of reading a file containing module directives (a scenario file) and of passing that information to a module.	Full	

MD5.1	This module shall be capable of reading a file one line at a time, extracting each line as a directive, and of passing the directive to another module.	Full	
MD5.2	This module shall allow for lines in the file, in addition to directive lines, of the following types: a. Comment, where the entire line is ignored b. Sleep, where the execution of the file is paused for an amount of time supplied in the line c. Start scenario, which would start a scenario based on a file pathname supplied in the line d. Conditional (IF & While clauses) execution within scenario script files.	Partial	a-c implemented
MD5.3	This module shall be capable of sending directives to various modules based on information supplied in the directive line: a. Regular set value directives b. Set container item with simple expression c. Set container item with other container items d. Boolean expressions e. With channel number (# num) specified at the beginning of the line.	Partial	a-d are implemented in the SIMSS Library. Unary negate operator may not work yet. If no channel number (# num) specified, the default is channel 1.
MD5.4	This module shall be capable of accepting from an external module, pathnames of scenarios to execute.	Full	Special formatted directives shall be used at the module connected upstream.
MD5.5	This module shall provide the user with the capability to indicate the files to read, up to a maximum of five files.	Full	
MD5.6	This module shall provide the user with the capability to stop, start, or pause file execution at any time.	Full	Generated scenarios can only be stopped by project stop
MD5.7	This module shall provide the user with status information including:	Full	

	a. The name of the file being read b. The current line number in the file c. The contents of the current line in the file d. Whether the module is running or stopped		
MD5.8	This module shall generate an event message for each directive line processed.	Full	Enabled by event message filtering from GUI

FD	***Flight Dynamics Requirements***	***R4.1 Impl.***	***Comments***
FD1	The SIMSS shall provide an interface to support flight dynamics modeling.		
FD1.1	The SIMSS shall be capable of supporting mission specific attitude modeling.		
FD1.2	The SIMSS shall be capable of supporting mission specific orbit modeling.		

SC	***Spacecraft Simulation Requirements***	***R4.1 Impl.***	***Comments***
SC1	The SIMSS shall provide a generic, data-driven module with the capability to receive and validate commands, create and send telemetry, reflect commands received in telemetry, and support data and subsystem modeling.		
SC1.1	This module shall fulfill all of the requirements for telemetry generation listed under either TG1 or TG2.		
SC1.2	This module shall fulfill all of the requirements for command ingest listed under either CI1 or CI2.		
SC1.3	This module shall be capable of supporting all of the timing requirements listed under TM1, TM2, TM3, and TM4.		
SC1.4	This module shall be capable of supporting all of the modeling requirements listed under MD1, MD2, and MD3.		
SC1.5	This module shall be capable of supporting all of the flight dynamics modeling requirements listed under FD1.		
SC1.6	This module shall be capable of using command verification information from the database to reflect valid commands received in telemetry.		

SC1.7	This module shall be capable of executing a predefined model or script in response to a command received or other spacecraft event.		
SC1.8	This module shall provide the user with the capability to select the source and version of the database to use for module operations.		

3.1	***System-Level Performance Requirements***	***R4.1 Impl.***	***Comments***
3.1.1	The SIMSS shall update status, data quality, and accounting information once every 10 seconds, at a minimum.	Full	For all modules implemented
3.1.2	The SIMSS shall acknowledge a request from a local user within 2 seconds of its entry.	Full	
3.1.3	The SIMSS shall start the execution of a local user request within 5 seconds of its entry.	Full	
3.1.4	The SIMSS shall be ready for operational use within 5 minutes of program execution exclusive of external dependencies.	Full	

3.2	***Serial Mode Performance Requirements***	***R4.1 Impl.***	***Comments***
3.2.1	The SIMSS shall be capable of supporting up to four channels consisting of one command and up to three telemetry streams.	Partial	One channel only
3.2.2	The SIMSS shall be capable of receiving or transmitting three simultaneous fixed block length data streams at data rates from a minimum of 100 bits per sec (bps) to a maximum rate of 2 Mbps.	Partial	Avtec card: 100 ~4M bps; ICS card: 900 ~2M bps
3.2.3	The SIMSS shall provide the capability to receive or transmit a single variable block length bit stream at data rates up to 192 Kbps.	Full	

3.3	***IP Mode Performance Requirements***	***R4.1 Impl.***	***Comments***
3.3.1	The SIMSS shall be capable of supporting up to four channels consisting of one command and up to three telemetry streams.	Full	
3.3.2	The SIMSS shall be capable of receiving or transmitting three simultaneous fixed block	Full	To be tested

	length data streams at data rates from a minimum of 100 bits per sec (bps) to a maximum rate of 2 Mbps.		
3.3.3	The SIMSS shall provide the capability to receive or transmit a single variable block length bit stream at data rates up to 192 Kbps.	Full	To be tested